AMENDMENTS TO THE CLAIMS

The listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A computer-implemented method for processing software code, said method comprising:

loading, at a second processor, a virtual machine program into a local memory corresponding to the second processor;

receiving, at [[a]] the second processor, a code processing request requested by a first processor, wherein the first and second processors are heterogeneous processors within a computer system that share a common memory;

reading, from the common memory shared by the first and second processors, software code data corresponding to the request, the software code data including virtual machine code adapted to be processed by the virtual machine program;

writing the software code data corresponding to the request to [[a]] the local memory corresponding to the second processor in response to the request; [[and]]

processing the software code data by the second processor[[.]], wherein the processing includes processing the virtual machine code at the second processor using the virtual machine program, the processing resulting in executable instructions;

writing the executable instructions to a memory location accessible by the first processor; and

executing, at the first processor, the executable instructions.

2. (Original) The method as described in claim 1 further comprising:

prior to the receiving:

reading script code from the common memory;

writing the script code to a local memory corresponding to the first processor;

interpreting, at the first processor, the script code, the interpreting resulting in the software code; and

writing the software code to the second processor's local memory.

3. (Original) The method as described in claim 1 further comprising:

writing data resulting from the executing to the common memory.

4. (Currently Amended) The method as described in claim 1 further comprising:

prior to the receiving:

running a first program; ~~, during the running of the first program, identifying a call to the software code; and~~

in response to running the first program, identifying a call to a software effect corresponding to the software code data; and

loading the software code data into the common memory, wherein the processing of the software code ~~is occurs simultaneously to~~ data occurs during the running of the first program and wherein the processing is completed prior to the ~~call of the software code from the~~ first program calling the software effect.

5. (Currently Amended) The method as described in claim 4 further comprising:

performing, by the second processor, a multimedia effect resulting from the processing of the software code data, ~~the performance performed by the second processor~~.

6. (Currently Amended) The method as described in claim 4 further comprising:

receiving, at the first processor, the executable instructions resulting from the processing performed by the second processor, wherein the executable instructions are adapted to perform a multimedia effect; and

performing the multimedia effect on the first processor by executing the received executable instructions.

7. (Currently Amended) The method as described in claim 1 ~~further comprising~~ wherein the writing further comprises:

~~loading, a the second processor, a virtual machine program into the second processor's local memory;~~

~~reading, from the common memory shared by the first and second processors, the software code data that includes virtual machine code adapted to be processed by the virtual machine program;~~

~~processing the virtual machine code at the second processor using the virtual machine program, the processing resulting in executable instructions;~~

writing the executable instructions to a memory location accessible by the first processor using a [[DMA]] direct memory access (DMA) operation[[; and]]

~~executing, at the first processor, the executable instructions~~.

8. (Original) The method as described in claim 7 wherein the memory location is selected from the group consisting of a local memory corresponding to the first processor, and the common memory.

9. (Currently Amended) The method as described in claim [[7]] 1 wherein the first processor has a first instruction set architecture and the second processor has a second instruction set architecture that is different from the first instruction set architecture, ~~and second processors are dislike processors with different instruction set architectures~~ and wherein the executable instructions resulting from the processing performed by the second processor are adapted to be executed on the first processor and not the second processor.

10. (Original) The method as described in claim 1 wherein the processing results in one or more program instructions adapted to be performed by the first processor, the method further comprising:

writing the program instructions to the common memory;

notifying the first processor that the program instructions have been written; and

executing the program instructions by the first processor.

11. (Currently Amended) An information handling system comprising:

a plurality of heterogeneous processors;

a common memory shared by the plurality of heterogeneous processors;

a first processor selected from the plurality of processors that sends a code processing request to a second processor, the second processor also being selected from the plurality of processors;

a local memory corresponding to the second processor;

a direct memory access (DMA) [[DMA]] controller associated with the second processor, the DMA controller adapted to transfer data between the common memory and the second processor's local memory; and

a processing tool for processing software code, the processing tool including software effective to:

load, at the second processor, a virtual machine program into the local memory corresponding to the second processor;

receive, at a second processor, the code processing request requested by the first processor;

read, from the common memory, software code data corresponding to the request, the software code data including virtual machine code adapted to be processed by the virtual machine program;

write the software code data corresponding to the request to the second processor's local memory in response to the request; [[and]]

process the software code data by the second processor[[.]], wherein the processing includes processing the virtual machine code at the second processor using the virtual machine program, the processing resulting in executable instructions;

write the executable instructions to a memory location accessible by the first processor; and

execute, at the first processor, the executable instructions.

12. (Original) The information handling system as described in claim 11 further comprising software code effective to:

prior to the reception of the request:

read script code from the common memory;

write the script code to a local memory corresponding to the first processor;

interpret, at the first processor, the script code, the interpreting resulting in the software code; and

write the software code to the second processor's local memory.

13. (Original) The information handling system as described in claim 11 further comprising software code effective to:

write data resulting from the executing to the common memory.

14. (Currently Amended) The information handling system as described in claim 11 further comprising software code effective to:

prior to the reception of the request:

run a first program; ~~, during the running of the first program, identify a call to the software code; and~~

identify a call to a software effect corresponding to the software code data in response to running the first program; and

load the software code data into the common memory, wherein the processing of the software code ~~is occurs simultaneously to~~ data occurs during the running of the first program and wherein the processing of the software code is completed prior to ~~the call of the software code from~~ the first program calling the software effect.

15. (Currently Amended) The information handling system as described in claim 14 further comprising software code effective to:

perform, by the second processor, a multimedia effect resulting from the processing of the software code data~~, the performance performed by the second processor~~.

16. (Currently Amended) The information handling system as described in claim 14 further comprising software code effective to:

receive, at the first processor, the executable instructions resulting from the processing performed by the second processor, wherein the executable instructions are adapted to perform a multimedia effect; and

perform the multimedia effect on the first processor by executing the received executable instructions.

17. (Currently Amended) The information handling system as described in claim 11 further comprising software code effective to:

~~load, a the second processor, a virtual machine program into the second processor's local memory;~~

~~read, from the common memory shared by the first and second processors, the software code data that includes virtual machine code adapted to be processed by the virtual machine program;~~

~~process the virtual machine code at the second processor using the virtual machine program, the processing resulting in executable instructions;~~

write, using a [[DMA]] direct memory access (DMA) operation, the executable instructions to a memory location accessible by the first processor[[; and]]

~~execute, at the first processor, the executable instructions~~.

18. (Original) The information handling system as described in claim 17 wherein the memory location is selected from the group consisting of a local memory corresponding to the first processor, and the common memory.

19. (Currently Amended) The information handling system as described in claim [[17]] 11 wherein the first processor has a first instruction set architecture and the

second processor has a second instruction set architecture that is different from the first instruction set architecture, ~~and second processors are dislike processors with different instruction set architectures~~ and wherein the executable instructions resulting from the processing performed by the second processor are adapted to be executed on the first processor and not the second processor.

20. (Currently Amended) The information handling system as described in claim 11 wherein the ~~process~~ processing results in one or more program instructions adapted to be performed by the first processor, the information handling system further comprising software code effective to:

write the program instructions to the common memory;

notify the first processor that the program instructions have been written; and

execute the program instructions by the first processor.

21. (Currently Amended) A computer program product stored on a computer operable media, the computer operable media containing instructions for execution by a computer, which, when executed by the computer, cause the computer to perform a method for processing software code, said ~~computer program product~~ method comprising:

loading, at a second processor, a virtual machine program into a local memory corresponding to the second processor;

~~means for~~ receiving, at [[a]] the second processor, a code processing request requested by a first processor, wherein the first and second processors are heterogeneous processors within a computer system that share a common memory;

reading, from the common memory shared by the first and second processors, software code data corresponding to the request, the software code data

including virtual machine code adapted to be processed by the virtual machine program;

~~means for~~ writing the software code data corresponding to the request to [[a]] the local memory corresponding to the second processor in response to the request; [[and]]

~~means for~~ processing the software code data by the second processor[[.]],wherein the processing includes processing the virtual machine code at the second processor using the virtual machine program, the processing resulting in executable instructions;

writing the executable instructions to a memory location accessible by the first processor; and

executing, at the first processor, the executable instructions.

22. (Currently Amended) The computer program product as described in claim 21 ~~further comprising~~ wherein the method further comprises:

prior to the ~~means for~~ receiving:

~~means for~~ reading script code from the common memory;

~~means for~~ writing the script code to a local memory corresponding to the first processor;

~~means for~~ interpreting, at the first processor, the script code, the interpreting resulting in the software code; and

~~means for~~ writing the software code to the second processor's local memory.

23. (Currently Amended) The computer program product as described in claim 21 ~~further comprising~~ wherein the method further comprises:

~~means for~~ writing data resulting from the executing to the common memory.

24. (Currently Amended) The computer program product as described in claim 21 ~~further comprising~~ wherein the method further comprises:

prior to the ~~means for~~ receiving:

~~means for~~ running a first program; ~~, during the running of the first program, identifying a call to the software code; and~~

in response to running the first program, identifying a call to a software effect corresponding to the software code data;

~~means for~~ loading the software code data into the common memory, wherein the processing of the software code ~~is occurs simultaneously to~~ data occurs during the running of the first program and wherein the processing is completed prior to ~~the call of the software code from~~ the first program calling the software effect.

25. (Currently Amended) The computer program product as described in claim 24 ~~further comprising~~ wherein the method further comprises:

~~means for~~ performing, by the second processor, a multimedia effect resulting from the processing of the software code data ~~, the performance performed by the second processor~~.

26. (Currently Amended) The computer program product as described in claim 24 ~~further comprising~~ wherein the method further comprises:

~~means for~~ receiving, at the first processor, the executable instructions resulting from the processing performed by the second processor, wherein the executable instructions are adapted to perform a multimedia effect; and

~~means for~~ performing the multimedia effect on the first processor by executing the received executable instructions.

27. (Currently Amended) The computer program product as described in claim 21 ~~further comprising~~ wherein the writing further comprises:

~~means for loading, a the second processor, a virtual machine program into the second processor's local memory;~~

~~means for reading, from the common memory shared by the first and second processors, the software code data that includes virtual machine code adapted to be processed by the virtual machine program;~~

~~means for processing the virtual machine code at the second processor using the virtual machine program, the processing resulting in executable instructions;~~

~~means for~~ writing the executable instructions to a memory location accessible by the first processor using a direct memory access (DMA) [[DMA]] operation[[; and]]

~~means for executing, at the first processor, the executable instructions~~.

28. (Original) The computer program product as described in claim 27 wherein the memory location is selected from the group consisting of a local memory corresponding to the first processor, and the common memory.

29. (Currently Amended)The computer program product as described in claim [[27]] 21 wherein the first processor has a first instruction set architecture and the second processor has a second instruction set architecture that is different from the first instruction set architecture, ~~and second processors are dislike processors with different instruction set architectures~~ and wherein the executable instructions resulting from the processing performed by the second processor are adapted to be executed on the first processor and not the second processor.

30. (Currently Amended) The computer program product as described in claim 21 wherein the ~~means for~~ processing results in one or more program instructions adapted to be performed by the first processor, the ~~computer program product~~ method further comprising:

~~means for~~ writing the program instructions to the common memory;

~~means for~~ notifying the first processor that the program instructions have been written; and

~~means for~~ executing the program instructions by the first processor.